

# **EXHIBIT 3**

# Exhibit 7

14023 U.S. PTO  
053106

PTO/SB/16 (10-05)

Approved for use through 7/31/2006. OMB 0651-0032

U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

**PROVISIONAL APPLICATION FOR PATENT COVER SHEET – Page 1 of 2**

This is a request for filing a PROVISIONAL APPLICATION FOR PATENT under 37 CFR 1.53(c).

Express Mail Label No. EV842148225US

INVENTOR(S)		
Given Name (first and middle [if any] )	Family Name or Surname	Residence (City and either State or Foreign Country)
Angelos D.	KEROMYTIS	New York, New York
Salvatore J.	Stolfo	Ridgewood, NJ
Additional inventors are being named on the _____ separately numbered sheets attached hereto		
TITLE OF THE INVENTION (500 characters max)		
Decoy Technology For Insider Threats In Large-Scale Networks		
Direct all correspondence to: CORRESPONDENCE ADDRESS		
<input checked="" type="checkbox"/> The address corresponding to Customer Number: <u>56949</u> OR <input type="checkbox"/> Firm or Individual Name		
Address		
City	State	Zip
Country	Telephone	Email
ENCLOSED APPLICATION PARTS (check all that apply)		
<input checked="" type="checkbox"/> Application Data Sheet. See 37 CFR 1.76 <input type="checkbox"/> CD(s), Number of CDs _____		
<input checked="" type="checkbox"/> Specification Number of Pages <u>18</u> <input type="checkbox"/> Other (specify) _____		
<input type="checkbox"/> Drawing(s) Number of Sheets _____		
Fees Due: Filing Fee of \$200 (\$100 for small entity). If the specification and drawings exceed 100 sheets of paper, an application size fee is also due, which is \$250 (\$125 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR 1.16(s).		
METHOD OF PAYMENT OF THE FILING FEES AND APPLICATION SIZE FEE FOR THIS PROVISIONAL APPLICATION FOR PATENT		
<input checked="" type="checkbox"/> Applicant claims small entity status. See 37 CFR 1.27.		
<input type="checkbox"/> A check or money order is enclosed to cover the filing fee and application size fee (if applicable). <u>100.00</u>		
<input type="checkbox"/> Payment by credit card. Form PTO-2038 is attached.		
<input checked="" type="checkbox"/> The Director is hereby authorized to charge the filing fee and application size fee (if applicable) or credit any overpayment to Deposit		
Account Number: <u>08-0219</u> A duplicative copy of this form is enclosed for fee processing.		

113374 U.S. PTO  
607899898  
053106

USE ONLY FOR FILING A PROVISIONAL APPLICATION FOR PATENT

Express Mail Label No. EV842148225US Dated: May 31, 2006

**PROVISIONAL APPLICATION COVER SHEET**

Page 2 of 2

PTO/SB/16 (10-05)

Approved for use through 7/31/2006. OMB 0651-0032

U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

The invention was made by an agency of the United States Government or under a contract with an agency of the United States Government.

☒ No.☐ Yes, the name of the U.S. Government agency and the Government contract number are:  
\_\_\_\_\_**WARNING:**

Petitioner/applicant is cautioned to avoid submitting personal information in documents filed in a patent application that may contribute to identity theft. Personal information such as social security numbers, bank account numbers, or credit card numbers (other than a check or credit card authorization form PTO-2038 submitted for payment purposes) is never required by the USPTO to support a petition or an application. If this type of personal information is included in documents submitted to the USPTO, petitioners/applicants should consider redacting such personal information from the documents before submitting them to the USPTO. Petitioner/applicant is advised that the record of a patent application is available to the public after publication of the application (unless a non-publication request in compliance with 37 CFR 1.213(a) is made in the application) or issuance of a patent. Furthermore, the record from an abandoned application may also be available to the public if the application is referenced in a published application or an issued patent (see 37 CFR 1.14). Checks and credit card authorization forms PTO-2038 submitted for payment purposes are not retained in the application file and therefore are not publicly available.

SIGNATURE \_\_\_\_\_

Date \_\_\_\_\_

May 31, 2006

TYPED or PRINTED NAME \_\_\_\_\_

Matthew T. Byrne

REGISTRATION NO.  
(if appropriate) \_\_\_\_\_

40,934

TELEPHONE \_\_\_\_\_

(212) 230-8800

Docket Number: \_\_\_\_\_

19240.00TBDUS10

Express Mail Label No. EV 842148225 US

## **DECOY TECHNOLOGY FOR INSIDER THREATS IN LARGE-SCALE NETWORKS**

We propose to develop a distributed “trap based” defense for detecting malicious threats attempting to propagate quietly throughout any network, including closed networks, and to begin bringing scientific discipline to *network* security application of well-published counter-intelligence techniques that have worked so well for so long in so many applications. The remainder of this section:

- Characterizes a number of threats of interest
- Provides additional context on current “state of art” in “trap based” network security defenses.
- Identifies specific claims regarding the outputs from this effort.
- Projects capabilities in context of the full spectrum of insider threats.

The increasing prevalence and sophistication of rootkits [11] and spyware [11, 30, 1] have elevated such malware to first-order security threats, particularly where snuck into closed networks, even if only very little information can be snuck back out. The ease with which such malware can be quickly inserted into a system through a variety of different vectors (*e.g.*, *USB key*, *CD*, files brought up from unclassified networks) and the many different ways such a threat can quietly propagate throughout a network (mail, web client/server interactions, database interactions, parallel use of authentication credentials) makes filtering-based prevention insufficient. This is particularly true for the intelligence community (IC) since the likely attack vector against the IC would be a slow and stealthy attack which is much more difficult to detect than “loud” fast-propagating worms [3, 4, 5, 6, 9, 17, 22]. Even with recent progress by “defense” in the arms race to the bottom of the software/hardware stacks [7, 21, 24], non-signature *behavior* based anomaly detection [35], it seems that a small set of adversaries backed by the resources and capabilities of the world’s most powerful competing nation-states might still find weaknesses through which to introduce such threats into closed networks. Given the emphasis placed on protecting the perimeter of such networks, through air-gaps, physical access control, and high assurance cross domain guards, once threats reach the relatively weaker “inside,” they can do tremendous damage, particularly if undetected for long periods of time as is often the case with insider threats.

Rather than continue to improve “preventative” defenses, where industry is spending billions of dollars already, and rather than continuing to improve anomaly and behavior based defenses, areas where both Columbia and Symantec already stand among the very best in the world, we propose to pursue a third alternative, “trap based” defenses, and to begin bringing scientific discipline to *network* security application of such techniques that have worked so well for so long in so many applications. Trap based defenses currently perform very well in “honeypots” and “honeyfarms” for trapping malware on the Internet, and industry is investing heavily in the scaling of such techniques. Industry is also investing heavily in translation of these techniques from the older server-side traps to the newer client-side “crawler” traps [36] which draw attacks from malicious servers, exposing the server’s capabilities and intent. Industry is also investing heavily in automating the processing of the somewhat surprising volume of information which these

Express Mail Label No. EV 842148225 US

traps generate. However, the critical fundamental limitations of these “traps” are their “lure” factor in drawing in more sophisticated threats, and realism or believability as “an environment worth targeting.” In short, existing traps don’t even pass a “first glance” test in emulating real targets. This is critical because “realism” is likely the single most important metric for projecting effectiveness of honeypots in trapping insider threats. For this reason, realism is the critical focus of the proposed research.

Specifically, we propose to develop and refine effectiveness of *active deception* at various semantic levels, by providing seemingly valid “bait” information and exposing this information as “bait traffic” within the closed network, between decoy accounts within the closed network. Bait traffic is pre-scripted and closely monitored in such a manner that any induced deviations are immediately noticed. A very tangible example, for environments with telnet, might be to dedicate set of machines to running prescribed telnet traffic between them with passwords exposed in the clear. The script for the traffic is known, so if anyone tries to “add” anything, this should be immediately and easily detected, regardless of whether the addition is simple login, injecting an attack into an already open session, or simply adding commands to an open session. Obviously the adversary succeeds against a machine intended to be disposable, and they are caught in the process. Of course, this seems unlikely to work in environments where telnet does not occur naturally, and overall, tremendous focus is given in this effort to automating creation of bait traffic that is perfectly realistic for nearly any individual specific environment of concern.

Our goals are to create entire “networks of bait traffic within the operational network” such that, even for periods of months following “first access” by a foreign intelligence entity, the foreign intelligence analysts are unaware that roughly half or more of the information in the environment is bait information, and even once aware of such traps, they cannot reliably discern original information from bait information. Our goals in creating such an infrastructure are to trick adversaries into attempting to leverage the trust relationships between decoy accounts to propagate between decoy accounts, or trick them into attempting to access bait information. It should be noted that propagation between decoy accounts within the network would require modifying the pre-scripted traffic between the accounts in such a manner that the deviations from script would be immediately noticed, and that any static stores of bait information can be similarly monitored for unscripted access. This system should be constructed in a manner that if an induced deviation is detected, the information made available from the system and the certainty of detection are sufficient for an actionable conclusion that the compartment has been compromised and appropriate steps must be taken. In such situations, compartments can be reconstituted elsewhere with different staff and different systems, and the original compartment might or might not blindly continue operation to determine more regarding the nature of the compromise, or the compartment may be dismantled immediately, or the compartment may be used to introduce disinformation. Such questions regarding how to handle any breach are strategic questions beyond the scope of the system. However, it is important to notice that the any determination of insider breach detection must have a very high certainty that is quite distant at the far end of the spectrum from the uncertainty of traditional “intrusion” detection systems.

What set of insider threats would such a “high realism,” “high certainty” insider breach detection traps address? If we consider insider threats categorized as below:

Express Mail Label No. EV 842148225 US

1. Insiders passively collecting with no effort to extend their social or network access.
2. Insiders actively socially networking for collection with or without any computer network.
3. Insiders who, wittingly or not and even if only very rarely and in very limited fashion, provide enemy access to the network for threats that extend their access strictly via the network.
4. Insiders inappropriately attempting to extend their access both socially and via the network.

In trapping any efforts to extend network access, the proposed effort directly addresses the third and fourth categories of threats. If successful, our work will result in a set of mechanisms and techniques that complement existing defensive strategies. To our knowledge, our approach is the *first* that integrates detection, deception and response to *targeted* malware threats. Although not a panacea, our approach should provide tremendously certain detection of any threat attempting to increase its access through the network, provide assistance in unnoticeable offline forensic analysis, and provide avenues for introduction of disinformation as needed. If such an infrastructure of “high realism and high certainty” traps is not created, the best resourced network insider threats will remain very, very difficult to detect, and uncertainty regarding integrity of each compartment will continue to grow as abilities and craft of network insider threats grows.

### III Technical Approach

Given the objectives above of creating a pervasive system of “high realism,” “high certainty” insider breach detection traps to reliably detect insiders inappropriately attempting to increase their network access, the main technical challenges in our project are:

1. Transparently recording all real information, events, and flows within the network to facilitate automated creation of bait information and bait traffic within the network. (Symantec)
2. Modeling recorded information and effectively transforming large volumes of information into bait information and scripts guiding execution of bait traffic, all with minimal human assistance. (Columbia)
3. Monitoring execution of bait traffic to detect induced deviations with high certainty, and recording context of any induced deviation with sufficient detail for confidence in forensic analysis. (Columbia)
4. Providing a secure and independent environment for the monitoring infrastructure. (Symantec)
5. Interleaving bait information and original information in the monitored system in such a manner that any user-inserted listening technologies cannot discern between “original” and “bait,” while preserving the property that bait information is never presented to a human user since we don’t want to confuse our own intelligence analysts. (Symantec)

In the remainder of this section, we discuss details of these primary tasks, and how our approach compares to other work in this space. We discuss the evaluation of our system in Section IV.

Our overall approach will be to have bait traffic emulate “in-host” and “on-network” behavior of original traffic. In fact, if the “in-host” behavior is emulated correctly, the hosts will invoke the network in precisely the same manner for the bait traffic as they did for the original traffic. For this reason, our focus is emulating “in-host” behavior in a manner that threats such as rootkits, malicious bots, keyloggers, and other spyware cannot discern the bait traffic from original traffic.

- a) A “surrogate user bot” (SUB) that appears to the operating system and all applications, and also all threats embedded in such, as if input were coming from a living, breathing, screen watching, keyboard and mouse using human user. Most commonly, the SUB will have a decoy account since this provides very clean separation of bait information from original information without introducing artifacts of handling bait information and original information differently within a single account. “Bridging” of real accounts and decoy accounts must be done very carefully, and is discussed at length further below.
- b) A virtualization layer beneath each operating system.
- c) An independent “on-host” monitoring environment.

Together, (b) and (c) facilitate an environment where the SUB can follow scripts to send events through virtualized keyboard and mouse drivers, opening applications, searching for messages, typing responses, surfing an intranet, cutting and pasting information, and doing anything else a normal user does, facilitated entirely by inputs through the virtualized keyboard and mouse drivers, guided entirely by the independent monitoring of the operating system and independent monitoring of the applications, user, and kernel space, with all results being displayed to virtualized screens, virtualized printers, and other virtualized output devices, with no information ever presented to a physical screen or physical printer. This environment also supports multiple SUB operating in parallel to each real human user, and the independent monitoring also supports the transparent recording of original information needed for generation of bait information. This independent monitoring will be done leveraging the SymEvent framework with hardware support of LaGrande and Vanderpool “secure virtualization” technologies.

COL00016194

Express Mail Label No. EV 842148225 US

environment to begin studying and refining realism of bait traffic, and studying the ability to closely monitor execution of scripts for “natural” network and host performance deviations from script, and also monitor for “induced” deviations from script.

### **III.2 Creating Realistic Bait Information**

Even with such a distributed infrastructure for deception with the network, increasing realism will be the cornerstone of progress. Bait traffic will include the set of content created within any host, the sequence of activities performed by users in any host, and proper characterization of how the users performed those activities.

Rather than apply use of bigrams [37] for generation of traffic, which results in “statistically” similar traffic, but traffic that does not pass a “first glance” test of legitimacy, we propose to replay “perfectly real” previously recorded information, and replaying that content some time after recording, and studying techniques for large scale removal of artifacts of the time delay, such as references to dates in any of the many formats which dates are casually included in free flowing text. Ideally, human readers should not be able to discern whether a set of messages are original content or bait content. Interestingly, this allows the problem to be posed as a purely scientific refinement challenge that can be tested with sets of messages, sets of refinement techniques, and sets of human reviewers. In fact, such tests can even be done with publicly available corpora of messages such as the ENRON corpus [38]. Interestingly, once automated techniques are perfected for translating recorded information into bait information, these techniques can be applied by any organization, and the techniques have two critical properties for any organization. First, the techniques do not expose information in any new environments, and the techniques do not expose any information that was not already previously exposed into the primary protected environment. Second, they provide bait that is tailored to the specific organization wishing to defend itself. This “self-tailored realism” is a critical piece lacking from current honeypots. Sale of commercial “honeypot-like defenses,” such as the Symantec Decoy Server, to enterprises, might be greatly improved if the honeypot more closely emulated the “target” wishing to defend itself. Such “tailored realism” then may be particularly valuable because commercial preventative defenses rapidly deploy very effective countermeasures for any “non-targeted” threats that wander into the industrial scale “non-tailored” threat collection honeypots.

Some bait traffic will be more difficult to replicate than others. For instance, email between decoy accounts transiting email servers shared with real accounts is relatively easy. However, where users post to “blog” style web-logs, it will be necessary to have a set of SUB recreating the blog on a decoy server that is not accessible to real users who do not go looking for it in inappropriate ways. There are other examples involving workflows and other shared workspaces which can be better addressed in the full proposal. Also, obviously some activities, like passwords and other authentication may also require update between first recording and replay. However, a critical point for all instances will be that the existence of such “attractive” information can carefully be made known through a controlled set of likely but inappropriate information gathering techniques expected to be used by threats wishing to collect information within closed networks.

Express Mail Label No. EV 842148225 US

### **III.3 Bridging from Real User Accounts into the Deception Infrastructure**

As mentioned above, it is important that threats not be able to discern the difference between bait and original information, or the difference between “decoy” and “real” servers and accounts, or the difference between real and SUB users. However, although the deception infrastructure will be attractively noticed by some threats attempting to listen with a large set of techniques, some threats may not notice or be willing to attempt propagation unless the deception infrastructure interacts with an account or operating system that they have compromised. Providing such “bridges” must be done carefully. For these cases, a SUB must be built to share a user account with a real user, and the virtualization and monitoring infrastructures must ensure that the real user is never presented bait information, unless they begin actively seeking it themselves through inappropriate techniques. A weak but simple means of doing this is to ensure that although the same account is used, the SUB and human user are never logged in at the same time, and that the SUB “cleans up after itself.” Alternatively, a different type of surrogate bot may be built to operate while the user is logged in, but only acting on background processes and acting in a manner that the user does not naturally notice. Examples might include browsers which once opened but “minimized” can be used to surf the intranet and returning to the user’s last page with history wiped without changing the minimized appearance of the browser. Similarly, minimized mail clients can send messages and delete sent messages, exposing interfaces to the “attractive” parallel network. However, doing such surgery without introducing artifacts suspicious to threats in the operating system and applications will require techniques similar to techniques for monitoring an operating system which would rather monitor itself without any “outside help.”

All in all, this will have to be done very carefully, and further elaboration is reserved for a full proposal.

### **III.4 Summary**

Emphasis of the first 18 months is focused on the critical hard problem of creating techniques for generating realistic bait on very large scale, creating the infrastructure for recording and playing back such volumes of traffic in a closely monitored manner, and bridging such a deception net into the production network. Focus of any continuing work would be improving the quality of deception, introducing techniques for transparent forensic collection, and developing interfaces for natural injection of disinformation in scale.

However, the core value proposition is the ability to effectively lure and detect a range of insider threats.

Express Mail Label No. EV 842148225 US

#### IV Evaluation Approach

Evaluation will focus on measuring *(in)distinguishability* (realism) of the bait information and bait environments. This will be evaluated at two levels:

- (a) First, corpora of messages will be translated to “bait” information, and the translation will be scored by the fraction of bait messages incorrectly identified as original messages, and fraction of people incorrectly identifying bait messages as original.
- (b) Later, in month 14, the system will be run in a “live” mode on a small closed test network with professional security consultants having deployed spyware, rootkits, keyboard loggers, and other sniffers to collect information to discern whether or not deceptive systems are present, or if all of the traffic is being generated by living breathing human users, and if present, determine if they can distinguish between the original and bait content. Results from this test should be available in month 15. This test will be run using mainly dedicated workstations with minimal usage of the “secure virtualization” hardware, with further integration of the secure virtualization hardware occurring in months 14 through 17.

To perform such “live” evaluation, we will use facilities both at Columbia’s Network Security and Intrusion Detection Labs (see appendix section on Organizational Capabilities) and Symantec’s large scale distributed performance testbeds. We will create synthetic traffic among these systems using long-term traffic traces from the Columbia Computer Science Department and other sources. Performers would additionally be happy to have such performance demonstrated in an IC Research and Development Experimental Collaboration (RDEC) testbed, the DETER testbed, or the IO Test Range. Given the sensitive nature of the most operationally relevant exercises, the RDEC and IO Test Range may have advantages over DETER.

We should note that we expect the “overhead” of such deception to be large, roughly doubling, or if desired, tripling the volume of traffic in the network to dramatically increase the likelihood that an insider would unwittingly target “bait” in addition to targeting legitimate systems. It should be noted however, that because the network is “shared” between legitimate and bait systems, administering the network for legitimate use effectively administers the network for the bait systems, so the network administration burden should not be increased, even though the volume grows substantially.

Express Mail Label No. EV 842148225 US

## Bibliography

The following references are hereby incorporated by reference herein in their entireties:

- [1] <http://www.cexx.org/adware.htm>.
- [2] <http://security.kolla.de/>.
- [3] CERT Advisory CA-2001-19: 'Code Red' Worm Exploiting Buffer Overflow in IIS Indexing Service DLL. <http://www.cert.org/advisories/CA-2001-19.html>, July 2001.
- [4] CERT Advisory CA-2001-26: Nimda Worm.  
<http://www.cert.org/advisories/CA-2001-26.html>, September 2001.
- [5] Cert Advisory CA-2003-04: MS-SQL Server Worm.  
<http://www.cert.org/advisories/CA-2003-04.html>, January 2003.
- [6] CERT Advisory CA-2003-21: W32/Blaster Worm.  
<http://www.cert.org/advisories/CA-2003-20.html>, August 2003.
- [7] W. A. Arbaugh. *Chaining Layered Integrity Checks*. PhD thesis, University of Pennsylvania, 1999.
- [8] William A. Arbaugh, David J. Farber, and Jonathan M. Smith. A secure and reliable bootstrap architecture. In *IEEE Security and Privacy Conference*, pages 65–71, May 1997.
- [9] M. Bailey, E. Cooke, F. Jahanian, D. Watson, and J. Nazario. The Blaster Worm: Then and Now. *IEEE Security & Privacy*, 3(4):26–31, July/August 2005.
- [10] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the Art of Virtualization. In *19<sup>th</sup> ACM Symposium on Operating Systems Principles (SOSP)*, October 2003.
- [11] J. Butler and S. Sparks. Spyware and Rootkits – The Future Convergence. *USENIX ;login.*, 29(6):8–15, December 2004.
- [12] N. Chou, R. Ledesma, Y. Teraguchi, and J. C. Mitchell. Client-Side Defense Against Web-Based Identity Theft. In *Proceedings of the ISOC Symposium on Network and Distributed Systems Security (SNDSS)*, February 2004.
- [13] M. Christodorescu and S. Jha. Static Analysis of Executables to Detect Malicious Patterns. In *Proceedings of the 12th USENIX Security Symposium*, pages 169–186, August 2003.
- [14] M. Christodorescu and S. Jha. Testing Malware Detectors. In *Proceedings of the International Symposium on Software Testing and Analysis (ISSTA)*, July 2004.
- [15] Weidong Cui, Vern Paxson, Nicholas C. Weaver, and Randy H. Katz. Protocol-Independent Adaptive Replay of Application Dialog. In *Proceedings of the 13<sup>th</sup> Symposium on Network and Distributed System Security (SNDSS)*, February 2006.
- [16] D. Dagon, X. Qin, G. Gu, W. Lee, J. Grizzard, J. Levine, and H. Owen. HoneyStat: Local Worm Detection Using Honeypots. In *Proceedings of the 7<sup>th</sup> International Symposium on Recent Advances in Intrusion Detection (RAID)*, pages 39–58, October 2004.

Express Mail Label No. EV 842148225 US

- [17] T. Dubendorfer, A. Wagner, T. Hossmann, and B. Plattner. Flow-Level Traffic Analysis of the Blaster and Sobig Worm Outbreaks in an Internet Backbone. In *Proceedings of the Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA)*, July 2005.
- [18] T. Garfinkel and M. Rosenblum. A Virtual Machine Introspection Based Architecture for Intrusion Detection. In *Proceedings of the Symposium on Network and Distributed Systems Security (SNDSS)*, pages 191–206, February 2003.
- [19] T. Holz. A Short Visit to the Bot Zoo. *IEEE Security & Privacy*, 3(3):76–79, May/June 2005.
- [20] Seung-Sun Hong and S. Felix Wu. On Interactive Internet Traffic Replay. In *Proceedings of the 8<sup>th</sup> International Symposium on Recent Advances in Intrusion Detection (RAID)*, pages 247–264, September 2005.
- [21] N. L. Petroni Jr., T. Fraser, J. Molina, and W. A. Arbaugh. Copilot -a Coprocessor-based Kernel Runtime Integrity Monitor. In *Proceedings of the 13<sup>th</sup> USENIX Security Symposium*, pages 179–194, August 2004.
- [22] D. Moore, C. Shanning, and K. Claffy. Code-Red: a case study on the spread and victims of an Internet worm. In *Proceedings of the 2nd Internet Measurement Workshop (IMW)*, pages 273–284, November 2002.
- [23] Niels Provos. A Virtual Honeypot Framework. In *Proceedings of the 13<sup>th</sup> USENIX Security Symposium*, pages 1–14, August 2004.
- [24] R. Sailer, X. Zhang, T. Jaeger, and L. van Doorn. Design and Implementation of a TCG-based Integrity Measurement Architecture. In *Proceedings of the 13<sup>th</sup> USENIX Security Symposium*, pages 223–238, August 2004.
- [25] S. Saroiu, S. D. Gribble, and H. M. Levy. Measurement and Analysis of Spyware in a University Environment. In *Proceedings of the ACM/USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pages 141–153, March 2004.
- [26] S. Smith. Magic Boxes and Boots: Security in Hardware. *IEEE Computer*, 37(10):106–109, October 2004.
- [27] J.D. Tygar and Bennet Yee. DYAD: A System for Using Physically Secure Coprocessors. Technical Report CMU-CS-91-140R, Carnegie Mellon University, May 1991.
- [28] A. Vasudevan and R. Yerraballi. Cobra: Fine-grained Malware Analysis using Stealth Localized-Executions. In *Proceedings of the IEEE Security & Privacy Symposium*, May 2006.
- [29] VMware, Inc. <http://www.vmware.com>.
- [30] T. J. Walsh and D. R. Kuhn. Challenges in Securing Voice over IP. *IEEE Security & Privacy Magazine*, 3(3):44– 49, May/June 2005.
- [31] J. Yang, C. Sar, P. Twohey, C. Cadar, and D. Engler. Automatically Generating Malicious Disks using Symbolic Execution. In *Proceedings of the IEEE Security &*

Express Mail Label No. EV 842148225 US

*Privacy Symposium*, May 2006.

- [32] Z. Ye, S. Smith, and D. Anthony. Trusted Paths for Browsers. *ACM Transactions on Information and System Security (TISSEC)*, 8(2):153–186, May 2005.
- [33] Bennet Yee. *Using Secure Coprocessors*. PhD thesis, Carnegie Mellon University, 1994.
- [34] V. Yegneswaran, P. Barford, and D. Plonka. On the Design and Use of Internet Sinks for Network Abuse Monitoring. In *Proceedings of the 7<sup>th</sup> International Symposium on Recent Advances in Intrusion Detection (RAID)*, pages 146–165, October 2004.
- [35] Nong Ye, A Markov Chain Model of Temporal Behavior for Anomaly Detection, In *Proceedings of the 2000 IEEE Workshop on Information Assurance and Security*, United States Military Academy, West Point, NY, 6-7 June, 2000.
- [36] Yi-Min Wang, Doug Beck, Xuxian Jiang, Roussi Roussev, Chad Verbowski, Shuo Chen, and Sam King, Automated Web Patrol with Strider HoneyMonkeys: Finding Web Sites That Exploit Browser Vulnerabilities, In *Proc. Network and Distributed System Security (NDSS) Symposium*, February 2006
- [37] Richard P. Lippmann, David J. Fried, Isaac Graf, Joshua W. Haines, Kristopher R. Kendall, David McClung, Dan Weber, Seth E. Webster, Dan Wyschogrod, Robert K. Cunningham, and Marc A. Zissman, "Evaluating Intrusion Detection Systems: the 1998 DARPA Off-Line Intrusion Detection Evaluation", In *Proceedings of the 2000 DARPA Information Survivability Conference and Exposition*, 2000, Vol. 2,
- [38] Bryan Klimt, Yiming Yang, Introducing the Enron Corpus, in *Proceedings of Third Conference on Email and Anti-Spam (CEAS 2006)*, July 27-28, 2006, Mountain View, California

APPENDIX A

Express Mail Label No:  
EV 842148225 US

The Double Cross System or XX System, was a World War II anti-espionage and deception operation of the British military intelligence arm; MI5. It involved capturing Nazi agents and using them to broadcast mainly erroneous information to the Nazi high command.  
—Wikipedia entry for “Double Cross”

## II Innovative Claims and Application(s)

The increasing prevalence and sophistication of spyware [11, 31, 1], such as network sniffers and keystroke loggers, have elevated such malware to first-order security threats. The ease with which such malware can be inserted into a system through a variety of different vectors (e.g., mail attachment, worm payload, web “drive-by” download) makes filtering-based prevention an insufficient defense. Furthermore, the likely attack vector against the IC would be a slow and stealthy attack, which is much more difficult to detect than “loud” fast-propagating worms [3, 4, 5, 6, 9, 17, 22]. Their ability to evade detection, which has resulted in an arms race to the bottom of the software/hardware divider, also raises concerns about the ability to detect them based on a signature or even based on *intrinsic* behavior (e.g., by monitoring system call sequences or memory region accesses). However, such malware is perhaps the number one threat in a computing environment that handles sensitive data and communications: exploitation of spyware-gleaned information, such as authentication credentials or server names/IP addresses, can lead to further compromise of other devices and services, leading to an avalanche compromise of the IT infrastructure on which the IC depends. Likewise, sensitive information contained in documents and files can be exfiltrated for later analysis. Unfortunately, detecting such data flows is not always possible, even in a relatively closed network environment.

We propose a complementary approach to detecting malware of various types that may be operating in an IC-like computing environment. Specifically, we propose *active deception* at various semantic levels, by providing seemingly valid but “booby-trapped” information. Our goals are threefold:

1. Detect the presence of malware and identify their likely locations inside the computing environment.
2. Assist in the identification of exfiltration channels and related data flows.
3. “Blind” the spyware by providing a possibly overwhelming amount of seemingly useful (but otherwise erroneous) information.

In this project, we will focus on two specific types of spyware: keystroke and password loggers, and network-activity sniffers. Our overall approach is to model user and host behavior (at the level of network flows, application use, and keystroke dynamics) and to create synthetic events conveying crafted information that will be difficult for spyware to differentiate from authentic events. Such events will be automatically generated and injected throughout the IT infrastructure using a variety of means, as background activity (*i.e.*, largely not noticeable by humans). In addition, such information can be produced out of band, under the guidance of IC officers. The crafted information (modeled after legitimate data flows and events) will be used to “steer” attackers that are seeking to make use of it towards decoy systems. The crafted information will be “personalized” to assist in the identification the compromised systems, and (in conjunction with network sniffers) to identify exfiltration channels and (possibly) destinations. The volume of such information (since it will be almost constantly generated) will also help to somewhat mask legitimate information that is captured in the same way. Finally, by steering attackers toward decoy systems, it is possible to feed them with various crafted information that can later be used for attribution, association analysis (who collaborates with whom and how information is shared among attackers), and countermeasures (although the latter is explicitly outside the scope of our work).

If successful, our work will result in a set of mechanisms and techniques that complement existing defensive strategies. To our knowledge, our approach is the *first* that integrates detection, deception and response to the whole malware threat. Although not a panacea, our approach enables active tracing and response to the *attempted exploitation* of exfiltrated information, and can assist in forensic analysis. If this work is not done, the IC community will remain vulnerable to the threat of slow, large-scale compromise with the only recourse being the participation in the purely defensive arms race defined by attempts at attack prevention and signature or behavior-based detection.

### III Technical Approach

Based on our high-level description of the proposed approach in the previous section, the main technical challenges in our project are:

1. Modeling of legitimate events and network flows, and creation of realistic deception information, both automatically and through human operator intervention.
2. Injection of this information in a surreptitious manner in the operating environment of potential spyware.
3. Detection of exploitation attempts through the use of decoy servers; the decoy servers may be dedicated, or they may be created on-demand on actual servers and workstations to create the illusion of a richer target environment.

Our overall approach is shown in Figure 1. In the remainder of this section, we discuss in more detail our technical plan of work, and how our approach compares to other work in this space. We discuss our evaluation plans in Section IV.

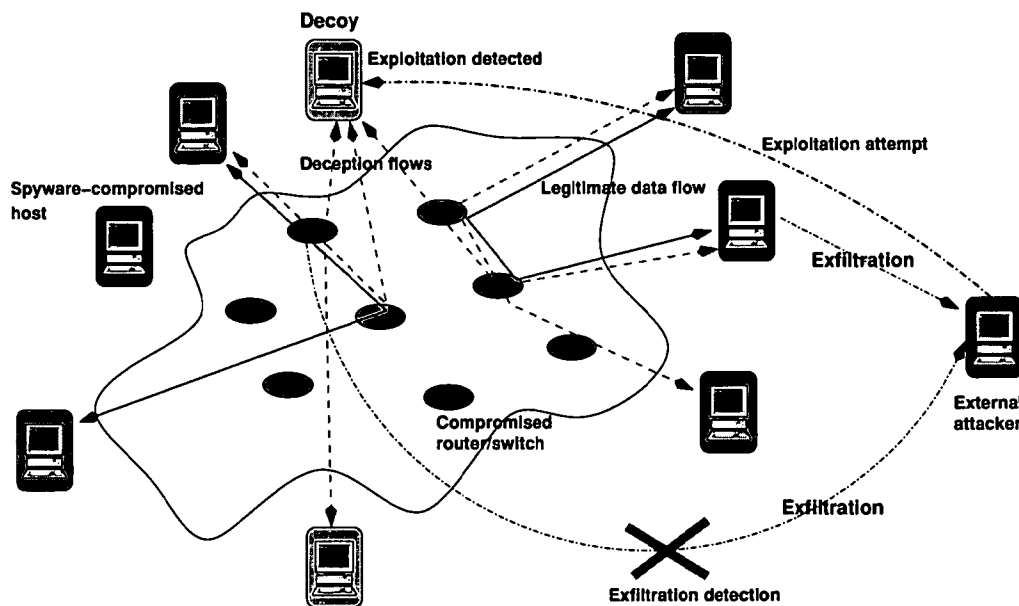


Figure 1: HIT-CI deception architecture. Deceptive flows (shown as dashed green lines), modeled after legitimate exchanges (shown as solid blue lines), are injected in the network by end-hosts and by special-purpose decoy stations. Similarly, keystrokes and application events are injected into end-hosts (not shown in figure). This information is caught by network and end-host sniffers and keyloggers, and is then exfiltrated to (external) nodes. The exfiltration itself may be detected through network monitoring. At a later point in time, the decoys detect the attempted exploitation of the deceptive information, identifying the compromised/malicious internal nodes. The decoy may continue the deception and identification of the remote attacker.

#### III.1 Traffic/Event Modeling

Given our focus on keystroke and password loggers (which operate on the end-host), and network sniffers (which may operate inside the infrastructure or at end hosts), we need to model two distinct types of information flows: network connections and keyboard events. With respect to the former, interesting information that an attacker may later attempt to exploit includes:

- the DNS name or IP address of the server (or, more generally, of the communicating peers);

Express Mail Label No: EV 842148225 US
---

- any authentication credentials captured (especially those that can be reused at a later time, such as a password);
- the data content of the flow, which may contain documents, email messages, or other information deemed interesting (*e.g.*, if the spyware is looking for specific keywords).

As we develop our prototype, it is likely that we will identify additional information, often protocol- and application-specific, that may be of interest to attackers. Our architecture allows us to easily incorporate additional deception modules that will exploit this information.

In the case of keyboard events that may be captured, passwords (or related authentication credentials) are the primary target and thus the information that we need to model.

**Network traffic modeling** We propose to use existing historical information (*i.e.*, previous network flows) as the basis for creating synthetic, traceable deception information. The network flows will be localized to specific machines and network segments, *e.g.*, we will not use a network flow captured in subnet A in a different subnet B to avoid exposing sensitive information that would otherwise remain hidden from a sniffer located in subnet B. A further advantage of this approach is that the traffic a particular sniffer sees will be very similar to prior traffic it has seen, thus mitigating the risk of detection.

The “personalization” of the deception traffic will be done across several different axis, following our previous identification of likely useful (and *actionable*) information that an attacker may collect.

**First**, we will replay data flows to servers with DNS names or IP addresses that are not typically used (*i.e.*, dedicated decoy servers). The replay will be a verbatim copy of the original data flow, with the exception of any protocol-dependent address or DNS translation that may be needed (relatively few protocols require this, and software modules for such translations are readily available commercially and in open-source products).

**Second**, we will replay data flows with different authentication credentials (username/password combinations). Again, we will use original data flows and client/server interactions (*e.g.*, a web-login and subsequent web page download), wherein the username and password will be modified to uniquely identify the flow, and thus the likely eavesdroppers (which will consist of the routers/switches and endpoint(s) of the flow). By cross-correlating subsequent exploitation attempts that use different sets of such credentials, we will be able to narrow down the list of possible leaks, potentially to a single machine or node.

**Third**, we will alter the data content of such flows to (a) make them more “interesting” to sniffers searching for particular keywords, and (b) modify the document content and/or structure to make it uniquely identifiable when it is transmitted over the network or if it is later seen on a different system (forensic analysis). The alteration can be made either through hard-coded rules (*e.g.*, injecting a “top secret” stanza on Word documents to make them more attractive) that are applied automatically, or be human-assisted. In the latter case, we envision the creation of a *Bait Information Creation Environment* (BICE), which will allow intelligence officers to create a rich collection of bogus information for use as data flows and by decoys. A means will be provided for CI officers to specify sets of changes to documents/data that are safe to use and sufficiently old, such that legitimate-looking documents will be generated that will allow for the tracking of the information. For example, dates, geographical locations, person names, *etc.* may be altered to create bait information. Steganographic or watermarking techniques may be used to embed identifying information in each data flow.

These three techniques will be used together to simultaneously make the flows more attractive to attackers, steer future exploitation attempts to the decoy server, and identify the source of the leaked credentials.

**Keyboard event modeling** Similar to the case of network traffic modeling, we propose to inject keyboard events in conjunction with application events that would indicate the typing of a username/password combination (*e.g.*, in a web browser window) or other interesting information (such as a URL to an enticingly named web server). Data modeling is simpler in this case, albeit the injection mechanism is somewhat more complicated as we describe next.

### III.2 Injection

Pursuant to our bimodal approach, we propose the use of two distinct data- and event-injection strategies.

On the network side, we will use decoy servers that both inject traffic and are the destinations of deceptive data flows. We will use readily available tools for traffic replay [20, 15], and the Symantec Decoy Server [?] as our prototype honeypot. In addition, we envision the use of dedicated workstations that are part of a wide-area deception

Express Mail Label No: EV 842148225 US
---

infrastructure, whose purpose will be the injection of deception traffic, supplied by BICE. These workstations may also serve as impromptu decoy servers (e.g., running lightweight monitoring sensors) or as repositories of tainted documents that are pointed at by other deception flows. Finally, as hardware virtualization techniques are becoming increasingly available in commodity systems, we anticipate the use of dedicated system images running side-by-side on actual end-user workstations that will subsume (or at least augment) the roles of centralized decoy servers and dedicated deception workstations. Ultimately, deception flows will be exchanged among all computing elements, achieving coverage of all components and potentially saturating the attacker's feeds. A signaling infrastructure will coordinate the transmission and reception of deception flows between the various HIT-CI components, such that only valid deception flows will appear to be accepted by the destination, to avoid mapping of the infrastructure by an active/probing adversary.

Keystroke injection is somewhat more complicated, due to the fact that such events will modify the state of the workstation, likely in undesirable ways. We propose to exploit advances in system virtualization to safely inject deception information to any keyloggers and other similar malware lurking on a user's workstation. Our approach is to periodically create snapshots of a user's environment and to replicate it inside a virtual machine [30, 10] that is not allowed to modify the persistent state (e.g., disk files). We will then create synthetic keyboard events from the host OS, which will be intercepted by any malware present on that user's system.

The VMs will be kept alive for a sufficiently long period of time to exfiltrate the captured information. As the processing requirements of an otherwise idle system image are low, we can easily migrate the system to a dedicated VM-hosting server. Outgoing communications from these may indicate information exfiltration. However, there are enough automated services in modern systems, including automatic updates, that this type of detection is likely to raise many false positives. Once virtualization becomes a true commodity service and all user software runs inside VMs by default, decoy VMs will run side-by-side with the user's active computing session(s) on the same workstation.

### III.3 Monitoring and Deception

On the monitoring side, we will use Symantec's Decoy Server

We also anticipate the use of lightweight deception mechanisms such as *honeyd* [24] on dedicated servers as well as end-nodes, to augment the deception infrastructure.

A companion (but independent) proposal has been submitted concerned with Accountable Document Flows and the Design and Development of an Infrastructure for Document Flow and Social Network Analysis. That proposal provides technology for monitoring information flows in a large scale network and the computation of descriptive statistical "profiles" describing those flows. The resultant statistical profiles has the dual purpose of providing the information necessary to create "believable but deceptive information flows" for the uses described herein. The additional effort required to generate the "decoy flows" is included in the statement of work of this proposal.

### III.4 Related Work

Much of the work relating to spyware defenses revolves around either preventing the installation of such malware or their subsequent detection and purging. The primary means through which this is achieved is signature- and, increasingly, behavior-based filtering, with a wide variety of commercially available tools. Some prominent such tools include Norton Antivirus (by Symantec), Windows Defender (by Microsoft), SpyBot [2], etc. At the network level, reverse and personal firewalls can detect some of the more obvious attempts at information exfiltration or malware "home-phone" attempts. More advanced detection approaches involve static disassembly and analysis [13, 14] or dynamic analysis [29, 35].

Anticipated hardware-based protection techniques involve the use of trusted modules that are used to isolate processes and provide a trusted path [27]. Application-specific spyware defenses have primarily focused on web-borne malware and password sniffers [12, 36].

The use of a virtual machine (VM) to identify the presence of rootkits was first proposed by Garfinkel and Rosenblum [18]. This is achieved by accessing the guest operating system low-level data structures, such as the list of active processes or operating system drivers, both through the guest operating system (where the results will be manipulated by the rootkit) and by directly accessing the same data structures from the host operating system. Any discrepancies when comparing the results would indicate the presence of a rootkit. Similar in concept to the previous item, the

Express Mail Label No: EV 842148225 US
---

use of specialized hardware (e.g., in the form of a PCI extension board) to monitor the state of kernel data structures [21]. Upon detection, the system may be rolled back to a known clean state or, if possible, disinfected. Other similar detection or “clean start” mechanisms that use hardware have been proposed over the years [8, 28, 25, 37, 7].

These techniques are important defensive weapons, but cannot be relied upon to always identify spyware. Furthermore, they are purely defensive tools—they can prevent the insertion of malware or aid in their removal, but do not exploit the malware to misinform and track the adversary.

Honeypots [24, 16, 38] are often used to detect the presence and behavior of bots [19] on the local network. They depend on scanning/attack behavior on the part of the attacker, and thus may not be entirely appropriate (or sufficient) against a more careful and methodical adversary. HoneyMonkeys [34] uses scripted web crawling with various browsers inside VMs to detect web sites that exploit browser vulnerabilities to surreptitiously install malware.

Saroiu *et al.* [26] conducted an analysis of spyware prevalence in an open academic environment. They discovered that 5% of the hosts in that network contained at least one (and often multiple) spyware. A study of various web sites [23] showed that 5.9% of the 18 million sites crawled contained “drive-by” downloadable spyware, and that 13.4% of the 21,200 executables found contained such malware.

## IV Evaluation Approach

We propose to build a prototype demonstrating the approach we outlined in the previous sections, and to then evaluate various aspects of the system. The work, the schedule for which is depicted in Figure 2, will be performed collaboratively at Columbia University and Symantec. Although we list evaluation as starting at 15 months, we expect that the individual components will be tested independently as they are being developed and completed.

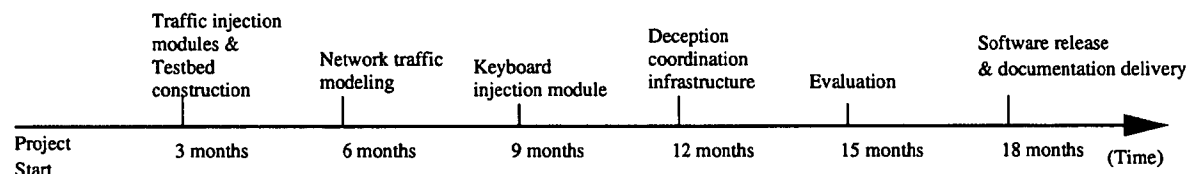


Figure 2: Milestones of our work across the 18 months of the project.

For our evaluation, we will measure three different (but related) aspects of the system: *overhead*, *coverage*, and *(in)distinguishability*. To perform this evaluation, we will use facilities both at Columbia’s Network Security and Intrusion Detection Labs (see appendix section on Organizational Capabilities) and at Symantec. Using these facilities, we will construct a virtual computing infrastructure composed of routers/switches and workstations (realized using VMs, most likely Xen and VMware, with remote desktop access through Windows Terminal Services or VNC). We will create synthetic traffic among these systems using long-term traffic traces from the Columbia Computer Science Department and other sources.

We will measure system *overhead* by determining the aggregate bandwidth consumed by the deceptions flows, as well as the induced network latency (and effective bandwidth decrease) seen by legitimate flows once our deception system prototype is in operation. We will measure *coverage* by deploying several types of relevant spyware available to us, suitably modified to report captured information to us, on “random” machines in the test network. We will conduct several experiments in which the deception system is not modified or otherwise steered by us subsequent to the introduction of the spyware in the system, to determine whether (and how well) we manage to inject bait information. Different types of spyware deployed at different points in the network will be used, to produce as many scenarios as possible. Finally, we will measure the *(in)distinguishability* of our deception information from “genuine” captured information by first aggregating “captured” information (i.e., the genuine credentials/documents caught by the spyware, and the deception information). We will then provide this information to members of our teams (and possibly students taking research projects or security classes at Columbia). We will not identify the genuine from the decoy information, and will ask them to find ways to distinguish the two. We anticipate the use of statistical and similarity tests such as n-gram analysis [32, 33]. A series of these experiments will allow us to refine our deception-generation techniques throughout the duration of the project.

Express Mail Label No: EV 842148225 US
---

## Bibliography

The following references are hereby incorporated by reference herein in their entireties:
---

- [1] <http://www.cexx.org/adware.htm>.
- [2] <http://security.kolla.de/>.
- [3] CERT Advisory CA-2001-19: 'Code Red' Worm Exploiting Buffer Overflow in IIS Indexing Service DLL. <http://www.cert.org/advisories/CA-2001-19.html>, July 2001.
- [4] CERT Advisory CA-2001-26: Nimda Worm. <http://www.cert.org/advisories/CA-2001-26.html>, September 2001.
- [5] Cert Advisory CA-2003-04: MS-SQL Server Worm. <http://www.cert.org/advisories/CA-2003-04.html>, January 2003.
- [6] CERT Advisory CA-2003-21: W32/Blaster Worm. <http://www.cert.org/advisories/CA-2003-20.html>, August 2003.
- [7] W. A. Arbaugh. *Chaining Layered Integrity Checks*. PhD thesis, University of Pennsylvania, 1999.
- [8] William A. Arbaugh, David J. Farber, and Jonathan M. Smith. A secure and reliable bootstrap architecture. In *IEEE Security and Privacy Conference*, pages 65–71, May 1997.
- [9] M. Bailey, E. Cooke, F. Jahanian, D. Watson, and J. Nazario. The Blaster Worm: Then and Now. *IEEE Security & Privacy*, 3(4):26–31, July/August 2005.
- [10] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the Art of Virtualization. In *19<sup>th</sup> ACM Symposium on Operating Systems Principles (SOSP)*, October 2003.
- [11] J. Butler and S. Sparks. Spyware and Rootkits – The Future Convergence. *USENIX ;login:*, 29(6):8–15, December 2004.
- [12] N. Chou, R. Ledesma, Y. Teraguchi, and J. C. Mitchell. Client-Side Defense Against Web-Based Identity Theft. In *Proceedings of the ISOC Symposium on Network and Distributed Systems Security (SNDSS)*, February 2004.
- [13] M. Christodorescu and S. Jha. Static Analysis of Executables to Detect Malicious Patterns. In *Proceedings of the 12th USENIX Security Symposium*, pages 169–186, August 2003.
- [14] M. Christodorescu and S. Jha. Testing Malware Detectors. In *Proceedings of the International Symposium on Software Testing and Analysis (ISSTA)*, July 2004.
- [15] Weidong Cui, Vern Paxson, Nicholas C. Weaver, and Randy H. Katz. Protocol-Independent Adaptive Replay of Application Dialog. In *Proceedings of the 13<sup>th</sup> Symposium on Network and Distributed System Security (SNDSS)*, February 2006.
- [16] D. Dagon, X. Qin, G. Gu, W. Lee, J. Grizzard, J. Levine, and H. Owen. HoneyStat: Local Worm Detection Using Honepots. In *Proceedings of the 7<sup>th</sup> International Symposium on Recent Advances in Intrusion Detection (RAID)*, pages 39–58, October 2004.

Express Mail Label No: EV 842148225 US
---

- [17] T. Dubendorfer, A. Wagner, T. Hossmann, and B. Plattner. Flow-Level Traffic Analysis of the Blaster and Sobig Worm Outbreaks in an Internet Backbone. In *Proceedings of the Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA)*, July 2005.
- [18] T. Garfinkel and M. Rosenblum. A Virtual Machine Introspection Based Architecture for Intrusion Detection. In *Proceedings of the Symposium on Network and Distributed Systems Security (SNDSS)*, pages 191–206, February 2003.
- [19] T. Holz. A Short Visit to the Bot Zoo. *IEEE Security & Privacy*, 3(3):76–79, May/June 2005.
- [20] Seung-Sun Hong and S. Felix Wu. On Interactive Internet Traffic Replay. In *Proceedings of the 8<sup>th</sup> International Symposium on Recent Advances in Intrusion Detection (RAID)*, pages 247–264, September 2005.
- [21] N. L. Petroni Jr., T. Fraser, J. Molina, and W. A. Arbaugh. Copilot - a Coprocessor-based Kernel Runtime Integrity Monitor. In *Proceedings of the 13<sup>th</sup> USENIX Security Symposium*, pages 179–194, August 2004.
- [22] D. Moore, C. Shanning, and K. Claffy. Code-Red: a case study on the spread and victims of an Internet worm. In *Proceedings of the 2nd Internet Measurement Workshop (IMW)*, pages 273–284, November 2002.
- [23] A. Moshchuk, T. Bragin, S. D. Gribble, and H. Levy. A Crawler-based Study of Spyware in the Web. In *Proceedings of the ISOC Symposium on Network and Distributed System Security (SNDSS)*, February 2006.
- [24] Niels Provos. A Virtual Honeypot Framework. In *Proceedings of the 13<sup>th</sup> USENIX Security Symposium*, pages 1–14, August 2004.
- [25] R. Sailer, X. Zhang, T. Jaeger, and L. van Doorn. Design and Implementation of a TCG-based Integrity Measurement Architecture. In *Proceedings of the 13<sup>th</sup> USENIX Security Symposium*, pages 223–238, August 2004.
- [26] S. Saroiu, S. D. Gribble, and H. M. Levy. Measurement and Analysis of Spyware in a University Environment. In *Proceedings of the ACM/USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pages 141–153, March 2004.
- [27] S. Smith. Magic Boxes and Boots: Security in Hardware. *IEEE Computer*, 37(10):106–109, October 2004.
- [28] J.D. Tygar and Bennet Yee. DYAD: A System for Using Physically Secure Coprocessors. Technical Report CMU-CS-91-140R, Carnegie Mellon University, May 1991.
- [29] A. Vasudevan and R. Yerraballi. Cobra: Fine-grained Malware Analysis using Stealth Localized-Executions. In *Proceedings of the IEEE Security & Privacy Symposium*, May 2006.
- [30] VMware, Inc. <http://www.vmware.com>.
- [31] T. J. Walsh and D. R. Kuhn. Challenges in Securing Voice over IP. *IEEE Security & Privacy Magazine*, 3(3):44–49, May/June 2005.
- [32] Ke Wang, Gabriela Cretu, and Salvatore J. Stolfo. Anomalous Payload-based Worm Detection and Signature Generation. In *Proceedings of the 8<sup>th</sup> International Symposium on Recent Advances in Intrusion Detection (RAID)*, pages 227–246, September 2005.
- [33] Ke Wang and Salvatore J. Stolfo. Anomalous Payload-based Network Intrusion Detection. In *Proceedings of the 7<sup>th</sup> International Symposium on Recent Advances in Intrusion Detection (RAID)*, pages 203–222, September 2004.
- [34] Y.-M. Wang, D. Beck, X. Jiang, R. Roussev, C. Verbowski, S. Chen, and S. King. Automated Web Patrol with Strider HoneyMonkeys: Finding Web Sites That Exploit Browser Vulnerabilities. In *Proceedings of the ISOC Symposium on Network and Distributed System Security (SNDSS)*, February 2006.

- 3